Express Virtual Channels: Towards the Ideal Interconnection Fabric

Amit Kumar†, Li-Shiuan Peh†, Partha Kundu‡ and Niraj K. Jha†
†Dept. of Electrical Engineering, Princeton University, Princeton, NJ 08544
‡Microprocessor Technology Labs, Intel Corp., Santa Clara, CA 95052
†{amitk, peh, jha}@princeton.edu, ‡partha.kundu@intel.com

ABSTRACT

Due to wire delay scalability and bandwidth limitations inherent in shared buses and dedicated links, packet-switched on-chip interconnection networks are fast emerging as the pervasive communication fabric to connect different processing elements in many-core chips. However, current state-of-the-art packet-switched networks rely on complex routers which increases the communication overhead and energy consumption as compared to the ideal interconnection fabric.

In this paper, we try to close the gap between the state-of-the-art packet-switched network and the ideal interconnect by proposing express virtual channels (EVCs), a novel flow control mechanism which allows packets to virtually bypass intermediate routers along their path in a completely non-speculative fashion, thereby lowering the energy/delay towards that of a dedicated wire while simultaneously approaching ideal throughput with a practical design suitable for on-chip networks.

Our evaluation results using a detailed cycle-accurate simulator on a range of synthetic traffic and SPLASH benchmark traces show upto 84% reduction in packet latency and upto 23% improvement in throughput while reducing the average router energy consumption by upto 38% over an existing state-of-the-art packet-switched design. When compared to the ideal interconnect, EVCs add just two cycles to the no-load latency, and are within 14% of the ideal throughput. Moreover, we show that the proposed design incurs a minimal hardware overhead while exhibiting excellent scalability with increasing network sizes.

Categories and Subject Descriptors: C.2.1 [**Computer Systems Organization**]: Network Architecture and Design - Packet-switching
General Terms: Design, Management, Performance
Keywords: Flow control, Packet-switching, Router design

1. INTRODUCTION

Driven by technology limitations to wire scaling and increasing bandwidth demands [1,2], packet-switched on-chip networks are fast replacing shared buses and dedicated wires as the *de facto* interconnection fabric in general-purpose chip multi-processors (CMPs) [3,4] and application-specific systems-on-a-chip (SoCs) [5–8]. While there has been significant work on interconnection networks for multiprocessors, design of on-chip networks, which face unique design constraints, is a relatively new research area. Ultra-low latency and scalable, high bandwidth communication is critical in on-chip communication fabrics in order to support a wide range of applications with diverse traffic characteristics. Moreover, these fabrics need to adhere to tight area and power budgets with tractable hardware complexity.

Modern state-of-the-art on-chip network designs use a modular packet-switched fabric in which network channels are shared over multiple packet flows. Even though this enables high bandwidth, it comes with a significant delay, energy and area overhead due to the need for complex routers. Packets need to compete for resources on a hop-by-hop basis while going through a complex router pipeline before traversing the output link at each intermediate node along their path. Thus, the packet energy/delay in such networks is dominated largely by contention at intermediate routers, resulting in a high router-to-link energy/delay ratio. In other words, the gap between current state-of-the-art networks and the ideal interconnect, in which all nodes are connected by pair-wise dedicated wires, is quite large.

In this work, we propose *express virtual channels (EVCs)*, a novel flow control and router microarchitecture design which tries to close the performance and energy gaps between the state-of-the-art packetized on-chip network and the ideal interconnection fabric. EVC-based flow control approaches the delay and energy of a dedicated link by allowing packets to *virtually* bypass intermediate routers along pre-defined virtual express paths between pairs of nodes. Thus, EVCs allow packets to skip the entire router pipeline at intermediate nodes and approach the energy/delay of a dedicated wire interconnect. Intuitively, this is achieved by statically designating a set of EVCs at each router that always connect nodes A and B that are k hops away, and prioritizing EVCs over normal virtual channels (NVCs) [9] at the intermediate nodes. For instance, Fig. 1(a) shows a 7×7 2D mesh network with three-hop EVCs ($k = 3$), with the dotted lines depicting EVCs, where EVCs are *not* additional physical channels, but virtual channels (VCs) [9] that share existing physical links. Traveling from node 00 to node 03 can then be done through an EVC which virtually skips the router pipelines at nodes 01 and 02. Fig. 1(b) shows an example of a typical packet route using EVCs where a packet traveling from node 01 to node 56 skips the router pipeline at nodes 04, 05, 16 and 26. Moving further, this work also proposes dynamic EVCs which use EVCs of varying lengths. With dynamic EVCs, packets at any node are allowed to choose among a range of EVCs and hop onto the one which is most suitable along their route towards their destination, thereby maximizing the use of EVCs.

In addition to lowering latency to that of a dedicated link, EVCs also reduce the amount of buffering and average router switching activity which makes them energy- and area-efficient. Moreover, as EVCs skip through arbitration at intermediate nodes, they reduce contention and push throughput. Hence, EVCs lead to savings in latency, throughput as well as energy, unlike prior work which tends to trade off one for the other (see Section 6 for a detailed discussion). Circuit switching [10] allows communications to approach the latency of dedicated wires (if the costly setup delay can be amortized) but, by dedicating physical

ISCA'07, June 9–13, 2007, San Diego, California, USA.

bandwidth to a message flow, suffers in throughput. Similarly, express cubes [11], in using physical express channels, trades off throughput when the express channels are not highly utilized. While recent work has proposed speculation [12, 13] to cut down the router critical path delay by parallelizing multiple pipeline stages, such techniques show diminishing returns with increasing network traffic when the speculation failure rate becomes high. EVCs, by *virtually* bypassing nodes in a non-speculative fashion, overcome these problems and are able to simultaneously approach the energy/delay/throughput of the ideal interconnection fabric.

In this paper, we first motivate the need for EVCs by highlighting the existing gap between current state-of-the-art packetized networks and the ideal network, both in terms of performance and energy consumption (Section 2). We then explain the details of EVCs and its dynamic variant in Section 3, before presenting the detailed microarchitecture and circuit schematics of the EVC router in Section 4. We evaluated EVCs using a cycle-accurate network simulator considering both synthetic traffic and traffic traces gathered from the execution of the SPLASH-2 benchmark suite [14]. Our results in Section 5 show upto 84% reduction in packet latency and upto 23% improvement in throughput while reducing the average router energy consumption by upto 38% as compared to an existing state-of-the-art packet-switched design. Section 6 contrasts EVCs with prior related work while Section 7 concludes the paper.

2. MOTIVATION

In this section, we present a motivating case study that highlights the latency-throughput performance and energy gap between the ideal interconnection fabric and an existing baseline design that incorporates several state-of-the-art router microarchitectural features that were recently proposed to tackle network latency, throughput and energy.

2.1 Baseline state-of-the-art router

Fig. 2(a) shows the microarchitecture of our baseline state-of-the-art VC router. For simplicity, we assume a two-dimensional mesh topology throughout this paper, though the router microarchitectures presented readily extend to other topologies. Thus, the router has five input and output ports corresponding to the four neighboring directions and the local processing element (PE) port. The major components, which constitute the router, are the input buffers, route computation logic, VC allocator, switch allocator and crossbar switch.

Fig. 3(a) shows the base router pipeline on which we will progressively add state-of-the-art router microarchitectural features. Since on-chip designs need to adhere to tight area budgets and low router footprints, we assume flit-level[1] buffering and credit-based VC flow control [9] at every router, as opposed to packet-level buffering. A head flit, on arriving at an input port, first gets decoded and buffered according to its input VC in the buffer write (BW) pipeline stage. In the next stage, the routing logic performs route computation (RC) to figure out the output port for the packet. The header then arbitrates for a VC corresponding to its output port in the VC allocation (VA) stage. Upon successful allocation of a VC, it proceeds to the switch allocation (SA) stage where it arbitrates for the switch input and output ports. On winning the output port, the flit then proceeds to the switch traversal (ST) stage, where it traverses the crossbar. This is followed by link traversal (LT) to travel to the next node. Body and tail flits follow a similar pipeline except that they do not go through RC and VA stages, instead inheriting the VC allocated by the head flit. The tail flit, on leaving the router, deallocates the VC reserved by the header.

To remove the serialization delay due to routing, prior work has proposed *lookahead routing (LA)* [15] where the route of a packet is determined one hop in advance, thereby enabling flits to compete for VCs immediately after the BW stage. Fig. 3(b) shows the router pipeline with lookahead routing. *Pipeline bypassing (BY)* is another technique that is commonly used to further shorten the router critical path by allowing a flit to speculatively enter the ST stage if there are no flits ahead of it in the input buffer queue. Fig. 3(c) shows the pipeline where a flit goes through a single stage of switch setup, during which the crossbar is set up for flit traversal in the next cycle while simultaneously allocating a free VC corresponding to the desired output port, followed by ST and LT. The allocation is aborted upon a port conflict. When the router ports are busy, thereby disallowing pipeline bypassing, aggressive *speculation (SP)* can be used to cut down the critical path [12,13,16]. Fig. 3(d) shows the router pipeline where VA and SA take place in parallel in the same cycle. If the speculation succeeds, the flit directly enters the ST pipestage. However, when speculation fails, the flit needs to go through these pipestages again depending on where the speculation failed.

The baseline router used in this study incorporates all the above-mentioned state-of-the-art techniques.

Router microarchitectural components: We next detail the microarchitectures we assumed for the different components within the baseline router. In order to make the design area- and energy-efficient, we assume single-ported buffers and a single shared port into the crossbar from each input. Separable VC and switch allocators modeled closely after the designs in [13] are assumed as they are fast and of low complexity, while still providing a reasonable throughput, making them suitable for the high clock frequencies and tight area budgets of on-chip networks. We also incorporate router microarchitectural optimizations for energy into our baseline design. First, *write-through input buffers* [17] are used, which save buffer read energy whenever a flit is able to directly bypass to the ST stage. Second, we adopt the *cut-through crossbar design* [17], which sacrifices the full connectivity provided by a matrix crossbar to reduce the area and energy overhead. In this work, we assume dimension-ordered routing for which a cut-through design shows no performance degradation due to reduced connectivity.

2.2 Ideal interconnection fabric

We next discuss the characteristics of the ideal interconnection fabric.

Ideal latency: A network with an ideal latency is one in which data travel on dedicated pipelined wires directly connecting their source and destination. The packet latency, T_{ideal}, in such a network is governed only by the average wire length D (the Manhattan distance) between the source and destination, packet size L, channel bandwidth b, and propagation velocity v:

$$T_{ideal} = D/v + L/b \quad (1)$$

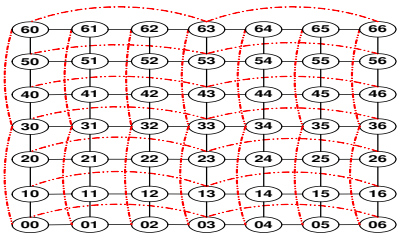
The first term corresponds to the time of flight which is the time spent traversing the interconnect, and the second to the serialization latency which is the time taken by a packet of length L to cross a channel with bandwidth b.

In a packet-switched network, the sharing and multiplexing of links between multiple source-destination flows result in increased packet transmission latency T, which is defined as the time elapsed between the first flit of the packet being injected at the source node to the last flit being ejected at the destination:

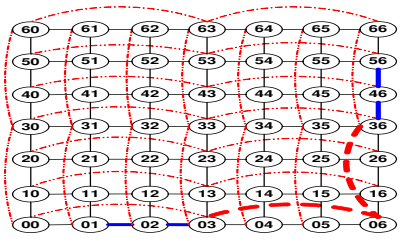
$$T = D/v + L/b + H \cdot T_{router} + T_c \quad (2)$$

where H is the average hop-count, T_{router} the delay through a single router, and T_c the delay due to contention [10]. The third term in Equation (2) corresponds to the time spent in the router coordinating the multiplexing of packets, while the fourth term is the contention delay spent waiting for resources. While a packet-switched network adds router pipeline and contention delay, the ideal network, however, in assuming that every tile is interconnected with every other tile, requires an enormous amount of global interconnect that has a detrimental effect on overall chip dimension, and thus D. In such a scenario, the wiring along the network bisection plane, which has the maximum number of wire tracks, forms the limiting factor. The chip edge length required to accommodate the total bisection wiring can be calculated as

[1] A flit is part of a packet, and the smallest unit of flow control. A packet consists of a head flit, followed by body flits, and ends with a tail flit.

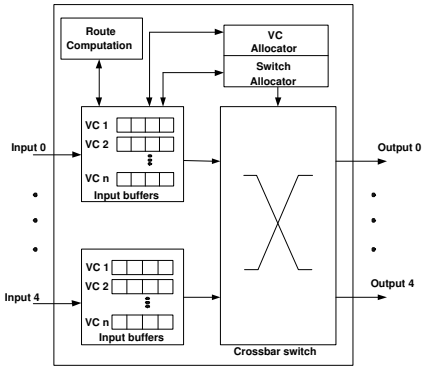


(a) An EVC network

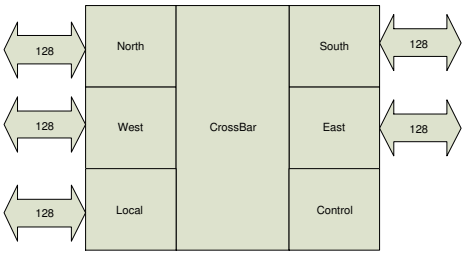


(b) VCs acquired from nodes 01 to 56

Figure 1: EVC network (solid lines are NVCs, dotted ones are EVCs)



(a) Baseline router microarchitecture



(b) Five-port router layout (input buffers are housed within the respective input ports, while VC, SA and control logic are grouped within **Control**)

Figure 2: Baseline router microarchitecture, design and layout

$$L_{edge} = 2 \cdot \frac{N}{2} \cdot \frac{N}{2} \cdot W_{pitch} \cdot c_{width} \quad (3)$$

where L_{edge} is the chip edge length, N the number of tiles, W_{pitch} the wire pitch and c_{width} the channel width.

We use the data from [18] to calculate the maximum wire length which can be driven in a cycle assuming delay-optimal insertion of repeaters and flip-flops. Assuming uniform placement of tiles, the average wire length D can then be derived as:

$$D = H \cdot \frac{L_{edge}}{\sqrt{N}} \quad (4)$$

It should be pointed out that such an ideal interconnection fabric, where each node has a dedicated interconnect to every other node, is not feasible in practice: a 7×7 network will require a chip size of $4760mm^2$ (L_{edge}=$69mm$), way beyond the ITRS projection [1] of a chip size of $310mm^2$ for high-performance chips.

Ideal energy: The energy consumption E_{ideal} of a packet in an ideal network is given by

$$E_{ideal} = L/b \cdot D \cdot P_{wire} \quad (5)$$

where D is the Manhattan distance between source and distance and P_{wire} the interconnect transmission power per unit length.

Again, multiplexing of packets in networks leads to additional energy consumption, with the energy E required to transmit a packet given by

$$E = L/b \cdot (D \cdot P_{wire} + H \cdot P_{router}) \quad (6)$$

where P_{router} is the average router power. P_{router} is composed of buffer read/write power, power spent in arbitrating for VCs and switch ports, and the crossbar traversal power [19].

Ideal throughput: Network throughput, which is defined as the data rate in bits per second that the network can accept per input port before saturation, is largely determined by the topology, flow control and routing mechanism. Given a particular topology, the ideal-throughput network is one which employs perfect flow control and routing to balance the network traffic over alternative paths while leaving no idle cycles on the bottleneck channels.

To study the ideal throughput, we simulated a network with unconstrained buffer capacity, unlimited VCs[2], and perfect switch allocation. Since this paper targets microarchitectural optimizations, we keep the topology and routing algorithm consistent between the baseline, ideal and proposed designs.

2.3 Existing gap

Here, we compare the state-of-the-art baseline design and ideal network in terms of latency, energy and throughput, noting the significant gap that still exists and motivating the need for further router microarchitectural innovations to bridge this gap.

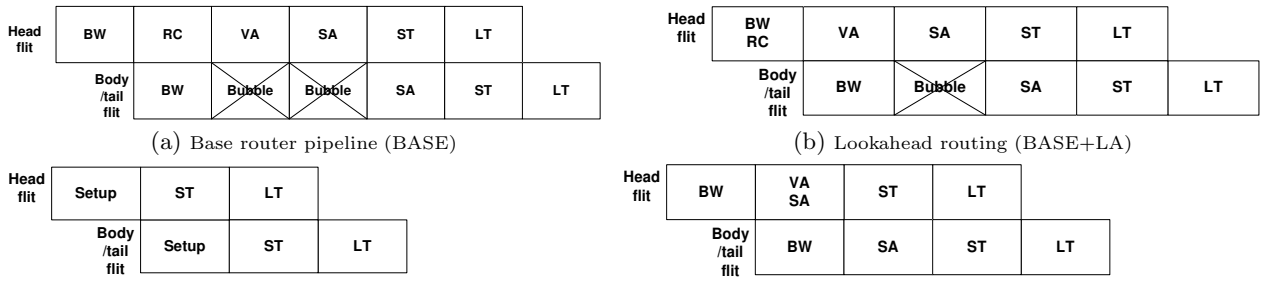
Simulation methodology: Table 1 in Section 5.1 shows the network parameters assumed for this study. Microarchitectural parameters, such as the number of VCs and buffers, were experimentally obtained by ensuring that they led to the best performance for the specific traffic pattern. Details of the simulation infrastructure are discussed in Section 5.

Design and pipeline sizing methodology: Fig. 2(b) shows the router layout we assumed throughout this paper. The buffers are laid out on the two sides of the router with the crossbar/switch in the center. In this layout, the height of the router is determined by the buffer height while the width of the router is determined by the wire pitch.

For each microarchitectural component, we design the circuit schematics to estimate the pipeline delay of each logical pipeline stage and size the router pipeline appropriately. The clock rate ($3GHz$ frequency) of the router, corresponding to a cycle time of 18 fanout-of-four (FO4) gate delays (excluding clock set up), was chosen based on being able to switch through the crossbar in a single cycle. The BW stage is used to write the flit into the buffers as well as set up all control signals in preparation for the next pipe stage (VA). It is also in the BW stage that we pre-compute the route (RC). The VA stage presents the longest critical path in our design and our modeling ascertains that the entire VA stage can be accommodated within a single 18FO4 cycle. The physical pipeline thus corresponds to that shown in Fig. 3.

Results: Fig. 4(a) compares the latency of the ideal network with that of the baseline network as a function of increasing network traffic. The latency of the ideal network,

[2]In actual simulations, this is mimicked by having very large numbers of buffers and VCs and verifying that a further increase in both resources does not lead to better performance.



(a) Base router pipeline (BASE)

(b) Lookahead routing (BASE+LA)

(c) Bypass pipeline (BASE+LA+BY)

(d) Speculative pipeline (BASE+LA+BY+SP)

Figure 3: Router pipeline [BW: Buffer Write, RC: Route Computation, VA: Virtual Channel Allocation, SA: Switch Allocation, ST: Switch Traversal, LT: Link Traversal]

which uses dedicated links between any pair of tiles, remains constant irrespective of the network load. For the baseline, we show the impact on network latency as we progressively incorporate lookahead routing, pipeline bypassing and aggressive speculation. Under very low network load ($< 15\%$ capacity), when most router ports are free, bypassing is able to significantly reduce packet latencies. Using speculation along with bypassing lowers the latency further with increasing load ($< 30\%$ capacity). However, as traffic increases, contention for network resources becomes higher, which leads to a higher probability of failed speculations and results in increasing latency until the network reaches the saturation point. Hence, it can be seen that router overhead leads to a significant latency gap between packet-switched networks and the ideal network.

Fig. 4(a) also highlights the significant throughput gap between the baseline and ideal-throughput network, with baseline's separable allocators only achieving 70% of the capacity of the ideal-throughput fabric.

Fig. 4(b) shows the gap between the energy consumption of the ideal network compared to the baseline network. Despite the baseline incorporating energy-efficient microarchitectural features, there still exists a substantial energy gap due to the additional buffering, switching and arbitration energy consumed in a router, which increases with network load until saturation is reached.

3. EXPRESS VIRTUAL CHANNELS: TOWARDS THE IDEAL INTERCONNECTION FABRIC

As explained in Section 2, the sharing and multiplexing of data on links in interconnection networks come at the cost of complex routers which contribute additional overhead in terms of packet latency and energy due to the router pipeline and resource contention while degrade throughput as a result of imperfect allocation of the limited bandwidth.

By virtually bypassing routers, EVCs, as proposed in this work, remove T_{router} and E_{router} at bypass hops, thus lowering energy/delay. As EVCs do not participate in arbitrations, they also lower T_c and improve allocation efficiency, thereby pushing throughput. Moreover, since the express paths created by EVCs are virtual as opposed to physical links, numerous EVCs of varying lengths can be used to connect nodes without the proportionate wiring area overhead. This allows packets to use EVCs which are tailored to their specific route and, hence, facilitates *dynamic adaptation* to different traffic patterns which further improves network performance and energy.

In the rest of this section, we explain the details of EVCs while Section 4 delves into the detailed microarchitectural design of the EVC-based router.

3.1 Static EVCs

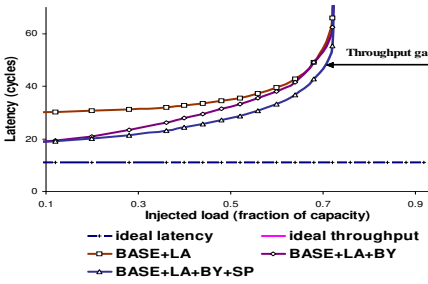
Here, we first present the details of EVCs using a static design which uses express paths of uniform lengths. All nodes in a static EVC network are distinguished as either an *EVC source/sink* node or a *bypass* node. A node is an EVC source/sink along a specific dimension if an EVC along that dimension originates/terminates at that node. Bypass nodes, on the other hand, do not act as sources and sinks of EVCs and are the ones which are virtually bypassed by packets traveling on EVCs. For example, in Fig. 1(a), node 00 is an EVC source/sink for both the X and Y dimensions while node 13 (04) is an EVC source/sink node along the X (Y) dimension. Nodes 01 and 02 (10 and 20) are examples of bypass nodes along the X (Y) dimension.

The entire set of VCs is divided into two types:

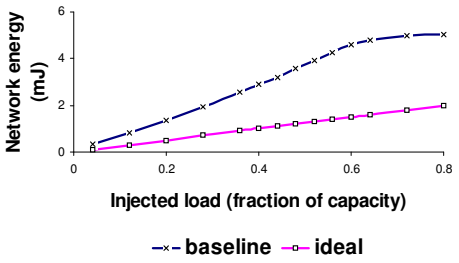
- *NVCs:* these are VCs which are allocated just like in traditional VC flow control [9] and are responsible for carrying a packet through a single hop.
- *k-hop EVCs:* these are VCs which carry the packet through k consecutive hops (where k is the fixed length of the EVC and is uniform throughout the network).

Bypass nodes only support the allocation of NVCs, not EVCs, with EVCs bypassing their router pipelines. Therefore, packets can acquire EVCs along a particular dimension only at EVC source/sink nodes. When a packet traveling on an EVC reaches a bypass node, it bypasses the entire router pipeline, skipping VC allocation as it continues on the same EVC it currently holds. It does not need to go through switch allocation as EVCs are prioritized over NVCs and are thus able to gain automatic passage through the switch without any contention. In other words, a packet traveling on a k-hop EVC traverses the next $k-1$ nodes without having to go through the router pipeline. Thus, a packet tries to traverse as many EVCs as possible along its route from the source to destination. NVCs are only used to reach an EVC source/sink in order to hop onto an EVC or when the hop-count in a dimension is less than k, the EVC length. Fig. 1(b) shown earlier depicts the VCs acquired by a packet traveling from node 01 to node 56 using deterministic XY routing. Here, the packet travels on NVCs from node 01 to 03, EVCs from nodes 03 to 06 and then 06 to 36, and finally NVCs from node 36 to 56. While connecting nodes using the virtual express lanes provided by EVCs, it should be noted that EVCs are restricted to connect nodes only along a single dimension and cannot be allowed to turn. Thus, packets are required to go through the router pipeline and change VCs when turning to a different dimension. This restriction is required to avoid conflicts between multiple EVC paths.

Impact on latency: Fig. 5(a) shows the *non-express* router pipeline for head, body and tail flits. Flits go through this pipeline at an EVC source/sink node or when they arrive at a bypass node on an NVC. The number of logical stages in this pipeline is identical to that in the baseline router (BASE+LA+BY+SP) described in Section 2. Fig. 5(b) shows the *express* router pipeline. A flit goes through this pipeline whenever it bypasses a node virtually, which happens when it arrives at a bypass node on an EVC. As a flit traveling on an EVC will continue on that EVC, there is no need to go through VA. It can also skip SA as EVCs are granted higher priority over NVCs and are automatically granted switch passage. Since no allocation is needed, an EVC flit can bypass BW and head directly to ST, followed by LT, to the next node at the end of which the flit gets latched. This pipeline, however, requires the switch to be set up *a priori*. We do this by sending a lookahead signal over a single-bit wire, which goes one cycle ahead to set up the



(a) Latency-throughput gap



(b) Energy gap

Figure 4: Existing gap between the state-of-the-art baseline and ideal interconnection fabric

switch at every intermediate hop which is bypassed, before the actual flit starts traveling on an EVC. Aggressive tailoring of the switch for EVCs can further shorten the *express* pipeline by removing the ST stage and allowing EVC flits to bypass the crossbar as well. Fig. 5(c) shows this pipeline. *As can be seen, the pipeline is now reduced to just link traversal: approaching that of the ideal interconnect.* Note that EVCs enable bypassing of the router pipeline at all levels of network loading, unlike prior techniques like bypassing or speculation which are effective only under low network load.

Impact on energy: Unlike speculation-based techniques, EVCs also lead to a significant reduction in network energy consumption. They do so by targeting the per-hop router energy E_{router}, which is given as:

$$E_{router} = E_{buffer\_write} + E_{buffer\_read} + E_{vc\_arb} + E_{sw\_arb} + E_{xb} \quad (7)$$

where E_{buffer_write} and E_{buffer_read} are the buffer write and read energy, E_{vc_arb} is the VC arbitration energy, E_{sw_arb} is the switch arbitration energy, and E_{xb} is the energy required to traverse the crossbar switch.

While traveling on an EVC, a packet is able to bypass the router pipeline of intermediate nodes, without the need for getting buffered or having to arbitrate for a VC or the switch port. *This in effect saves E_{buffer_write}, E_{buffer_read}, E_{vc_arb} and E_{sw_arb}, thereby significantly reducing E_{router} and approaching ideal energy.* Moreover, since bypass nodes support only NVCs and do not buffer EVC flits, the total amount of buffering is reduced, which leads to a corresponding reduction in energy consumption and area. The aggressive express pipeline removes E_{xb} as well, though wire energy E_{wire} increases slightly because of higher load (see Section 4).

Impact on throughput: Given a particular topology and routing strategy, network throughput is largely determined by the flow control mechanism. A perfect flow control is one which makes efficient use of network resources, leaving no idle cycles on the bottleneck channels. Using virtual express lanes, which effectively act as dedicated wires between pairs of nodes, EVC-based flow control is able to create partial communication flows in the network, thereby improving resource utilization and reducing contention T_c at individual routers. *Thus, packets spend less time waiting for resources at each router which lowers the average queuing delay, allowing the network to push through more packets before saturation and hence approach ideal throughput.*

3.2 Dynamic EVCs

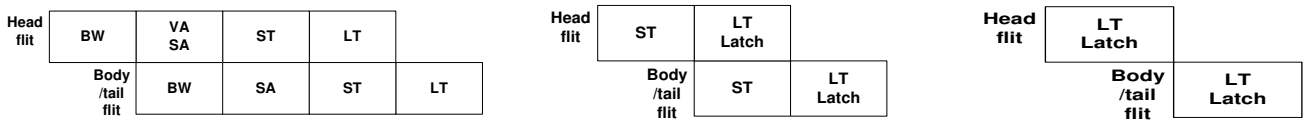
Using static EVCs of a fixed uniform length to connect source/sink nodes throughout the network, as discussed in the previous section, results in a constrained and asymmetric design. Firstly, the classification of all nodes as bypass and source/sink leads to an asymmetry in the design. Packets originating at bypass nodes are forced to first travel on NVCs before they can acquire an EVC at a source/sink node. Moreover, static EVCs lead to a non-optimal usage of express paths when packet hop-counts do not match the static EVC length k. In this case, packets end up bypassing fewer nodes along their route. In other words, static EVCs are biased towards traffic originating at source/sink nodes and with hop-count equal to or a multiple of the EVC length. Dynamic EVCs overcome these problems by: (a) making every node in the network a source/sink, and (b) allowing EVCs of varying lengths to originate from a node. By making all nodes identical, dynamic EVCs lead to a symmetric design ensuring fairness among nodes. Moreover, instead of having a fixed length, EVC lengths are allowed to vary between two hops upto a maximum of l_{max} hops. This improves adaptivity by allowing packets to pick EVCs of appropriate lengths to match their route the best. Fig. 6(a) shows dynamic EVCs along a particular dimension with l_{max} = 3. As can be seen, each node acts as a source/sink of EVCs of lengths two and three hops. Fig. 6(b) shows the VCs acquired by a packet traveling from node 01 to node 56 using XY routing in a network with l_{max} = 3. Since all nodes are source/sink nodes, the packet can acquire an EVC at its source node 01, taking the longest possible EVC (three-hop) to reach node 04. It then takes a two-hop EVC to reach node 06, followed by traversing a three-hop EVC in the Y dimension to reach node 36. Finally, the packet takes a two-hop EVC to reach its destination node 56. It can be seen that as compared to static EVCs (Fig. 1(b)), dynamic EVCs can adapt to the exact route of the packet, thereby allowing it to bypass more nodes and, hence, result in better performance and energy characteristics.

Similar to static EVCs, dynamic EVCs are restricted to be along a single dimension to prevent conflicts. It should be noted that overlapping of multiple EVC paths along the same dimension is allowed since all overlapping EVCs use network links in a sequentially ordered fashion. From a node's perspective, it always prioritizes any EVC flits it sees over locally-buffered flits.

Implementation of dynamic EVCs requires partitioning the entire set of VCs at any router port between NVCs and EVCs of lengths from two through l_{max} hops. Thus, unlike static EVCs, which partition all VCs between two bins of NVCs and uniform-length EVCs, dynamic EVCs divide the VCs into a total of l_{max} bins. A simple scheme is to uniformly partition all VCs by allocating an equal number of them to each bin.

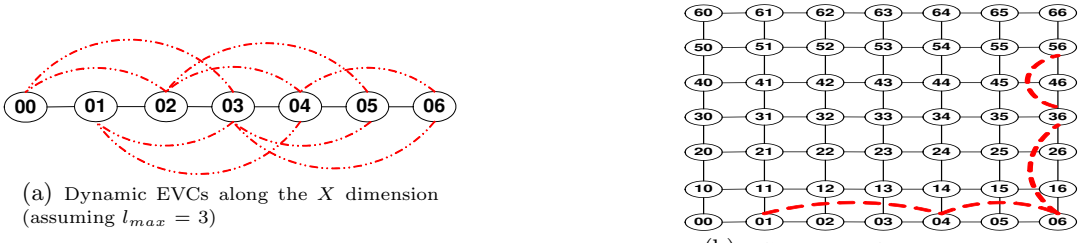
Routing flexibility using dynamic EVCs: The performance of dynamic EVCs can be further improved by allowing for flexibility in EVC traversals. While, normally, a packet tries to acquire the longest possible EVC along its path in order to bypass the maximum number of nodes, this scheme can be relaxed under certain scenarios when contention for EVCs of a particular length is high. For instance, consider a scenario where a packet has to travel p hops in a particular dimension, where $p \leq l_{max}$. In this case, if a p-hop EVC is not available but a smaller-length EVC is free, the packet can choose to switch to a smaller-length EVC. Effectively, this implies that longer EVC traversals can be broken down into a combination of shorter EVC/NVC traversals in order to spread the traffic between all virtual paths and, hence, reduce contention.

In order to make effective use of the routing flexibility in EVCs, the VC pool partitioning should be made *non-uniform*, with more VCs allocated to virtual paths of smaller lengths. This is because longer EVCs can only be used by packets with larger hop-counts and will remain unutilized if the traffic pattern has shorter distances to travel. On the other hand, by allocating more VCs to shorter paths,



(a) Non-express pipeline (b) Express pipeline (c) Aggressive express pipeline

Figure 5: EVC router pipelines



(a) Dynamic EVCs along the X dimension (assuming $l_{max} = 3$)

(b) VCs acquired from nodes 01 to 56

Figure 6: Dynamic EVC network

long-distance traffic always has the option to switch to a combination of shorter EVCs if contention for longer EVCs is high.

3.3 EVC buffer management

Buffered flow control techniques require a mechanism to manage buffers and communicate their availability between routers. Since a k-hop EVC creates a virtual lane between nodes which are k hops apart, it assumes buffer availability information to be communicated across k hops so that flits are ensured of a free buffer at the downstream EVC sink node. In this section, we present two schemes for buffer management in an EVC-based network.

Statically-managed buffers: This scheme partitions the buffer pool at a given router port between all VCs, statically reserving a set of buffers for each VC. Credit-based flow control [10] is used in which the upstream router maintains a count of the number of free buffers available downstream. This count is decremented each time a flit is sent out, thereby consuming a downstream buffer. On the other hand, when a flit leaves the downstream node and frees its associated buffer, a credit is sent back upstream and the corresponding free buffer count is incremented.

The number of buffers reserved for each VC is based on the corresponding credit round-trip delay. Fig. 7(a) shows the timeline for buffer management for NVCs. Using the timeline, the NVC credit round-trip delay, $T_{crt,n}$ can be calculated as:

$$T_{crt,n} = 1 + p + 1 + r = p + r + 2 \quad (8)$$

Equation (8) assumes that it takes one cycle for the credit to travel upstream, p cycles for flit processing at the upstream node, one cycle for the flit to travel downstream and r is the number of non-express router pipeline stages.

Fig. 7(b) shows a similar timeline for EVCs. Each k-hop EVC is allocated a minimum of $T_{crt,e}$ buffers:

$$T_{crt,e} = c + p + 1 + 1 + 2(k-1) + r = c + p + 2k + r \quad (9)$$

assuming it takes c cycles for the credit to propagate upstream to the EVC source, p cycles for flit processing, one cycle for the lookahead signal, one cycle for the flit to traverse the link to reach the first node which is to be bypassed, $2(k-1)$ cycles for the flit to propagate to the downstream EVC sink by bypassing $k-1$ intermediate nodes (assuming the express router pipeline), and r cycles to traverse the non-express router pipeline at the downstream EVC sink. As $T_{crt,e}$ is larger than $T_{crt,n}$, EVCs require deeper buffers than NVCs to cover the credit round-trip delay. Note that $T_{crt,e}$ can be reduced by using upper metal layers for credit signaling, thereby making $c < k$. In this paper, however, to minimize overhead and to be consistent with the baseline, we kept credit signals on the same metal layer, thus $c = k$. Moreover, $T_{crt,e}$ also decreases when using the aggressive express pipeline due to a shorter bypass pipeline.

While static buffer management is easy to implement, it is inefficient in allocating buffers in case of adversarial traffic. Consider the case when the majority of network traffic is nearest-neighbor, i.e., each node is communicating only with its immediate neighbor. In this case, EVCs are never used and the buffer space statically assigned to EVCs goes unutilized. Moreover, with $T_{crt,e}$ being directly proportional to k, longer EVCs require more buffers. This overhead can be large, especially in the case of dynamic EVCs where each node sources and sinks multiple EVCs of varying lengths.

Dynamically-managed buffers: To reduce buffer overhead and improve utilization, dynamic buffer management can be used in which buffers at each router are shared between all VCs. In this scheme, each router port maintains a pool of free buffers which can be allocated to any incoming EVC or NVC flit. Flow control is performed using threshold-based management where a node sends a *stop token* to an upstream node to which it is connected (either virtually through EVCs or physically through NVCs) when the number of free buffers falls below a pre-calculated threshold, Thr_k. On receiving this token, the upstream node stops sending flits to the corresponding downstream node. Conversely, as downstream buffers are freed and their number exceeds Thr_k, a *start token* is sent upstream to signal restart. To ensure freedom from deadlocks, one buffer slot is reserved for each VC. This ensures that a packet can make progress even if there are no buffers left in the free pool.

The value of Thr_k is calculated based on the hop distance between communicating nodes, which for a virtual path depends on the corresponding EVC length k, and is given by:

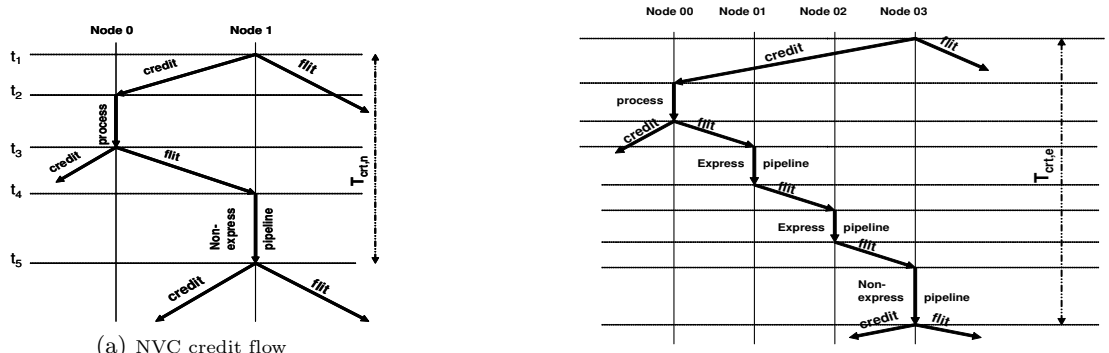
$$Thr_k = c + 2k - 1 \quad (10)$$

where c is the number of cycles taken by the token to propagate to the upstream EVC source while there can be a maximum of $2k-1$ flits in-flight which have already left the EVC source and need to be ensured of a buffer downstream (assuming the express pipeline and one cycle for token processing). Again, the value of Thr_k decreases when using the aggressive express pipeline because of a shorter bypass pipeline and correspondingly lower number of in-flight flits and even though global wires can be used to reduce c, in this work we assume $c = k$. For NVCs, the threshold can be calculated by setting k to one. In case of dynamic EVCs, multiple threshold values are used corresponding to each EVC length, with longer EVC paths being turned off first followed by smaller EVCs, and finally NVCs, whereas the reverse happens as buffers become free.

To overcome the buffer overhead of static buffer management, we use dynamically-managed buffers in this work.

3.4 Starvation avoidance

In any network, which pre-reserves bandwidth for specific message flows, a starvation scenario may arise when messages traveling on a pre-established circuit block other messages. Similarly, in the case of a network employing EVCs, the higher priority given to EVC flits can lead to a starvation scenario. More specifically, if a node along the path of an EVC always has incoming EVC flits to service, flits



(a) NVC credit flow

(b) EVC credit flow (for a three-hop EVC)

Figure 7: Credit round-trip delay with EVCs

buffered locally at the node may never get a chance to use the physical channel.

A typical starvation scenario for static EVCs is depicted in Fig. 8(a), where NVC flits buffered to go out at the east output port of node 03 are starved by the EVC (shown in bold). One simple algorithm to avoid such scenarios involves each bypass node maintaining a count of the number of consecutive cycles for which it has served an EVC. After serving EVC flits for n consecutive cycles, a bypass node sends a *Starvation on* token upstream to the EVC source node if it has NVC flits which are getting starved. Upon receiving this token, the EVC source node stops sending EVC flits on the corresponding link for the next p consecutive cycles. Hence, starved locally-buffered NVC flits can now be serviced. n and p are design parameters which can be set empirically.

In case of dynamic EVCs, starvation signaling needs to be done across more than one EVC source as multiple EVC paths may be bypassing a node. Fig. 8(b) shows a typical starvation scenario (for a network with $l_{max} = 3$), where flits buffered to go out at the east output port of node 03 get starved by the bypassing EVC traffic. EVC flows which can potentially lead to this starvation scenario are depicted in bold. The starvation avoidance algorithm used is similar to the one used for static EVCs with every node maintaining a count of the number of consecutive cycles for which it has served EVC flits. When this count exceeds a threshold n and if that node has locally-buffered flits waiting to use the physical channel, it sends out a *Starvation on* token. Considering the example in Fig. 8(b), when the count at node 03 exceeds n, it sends such a token to the neighboring node 02 which stops sending locally-buffered EVC flits (if any) along its east output port for the next p cycles. Node 02 then forwards the *Starvation on* token to its next neighbor which then stops sending EVC flits on its east output port as well. In general, forwarding of *Starvation on* tokens is done for $l_{max} - 1$ hops to cover all potential bypassing EVC paths. Moreover, to reduce the wiring overhead of starvation signaling, if more than one node is getting starved simultaneously, *Starvation on* tokens can be merged. For instance, if node 02 in the above example receives a *Starvation on* token from node 03 and detects starvation for its local flits at the same time, it can merge its own *Starvation on* token with the existing token (instead of creating a new token), and send it for forwarding to the next $l_{max} - 1$ hops (upto node 00).

Previously proposed deadlock-avoidance techniques can be readily used with EVCs. For instance, escape routes [10] can be created using only NVCs (similar to traditional VC flow control). Even though NVCs have a lower priority compared to EVCs, they are guaranteed to make progress using the starvation-avoidance algorithm of the EVC design.

4. EVC ROUTER MICROARCHITECTURE DESIGN

Fig. 9 shows the microarchitecture of a router in an EVC-based design. The differences from a generic router are shaded. For a *bypass* node in the static EVC design, the changes include the VC allocator, which now only handles NVCs, and the EVC latch which holds the flit arriving on an EVC. The crossbar switch can remain unchanged, or can also be aggressively designed to bypass ST as well for EVC flits. On the other hand, for a *source/sink* node in the static EVC design, the entire set of VCs at each input port is divided between EVCs and NVCs, each with a separate allocator. The EVC latch is not required as flits are not allowed to bypass source/sink nodes. In case of dynamic EVCs, nodes are not classified as bypass and source/sink. All nodes have the same microarchitecture, with the differences from a generic router including the EVC latch, division of VCs at each port into EVCs and NVCs (with a separate allocator for each) while the crossbar switch can again remain unchanged or aggressively designed. Since connections between non-adjacent nodes in an EVC design are virtual as opposed to physical, it can be seen that an EVC design does not require any additional router ports as compared to the baseline (even though each node may be connected to many other nodes using multiple express connections), thereby imposing no siginificant increase in the area/energy overhead of individual routers.

We next detail the design of each microarchitectural component of the EVC-based router. Note that the same design and pipeline sizing methodology we used for the baseline router is applied to that of the EVC router.

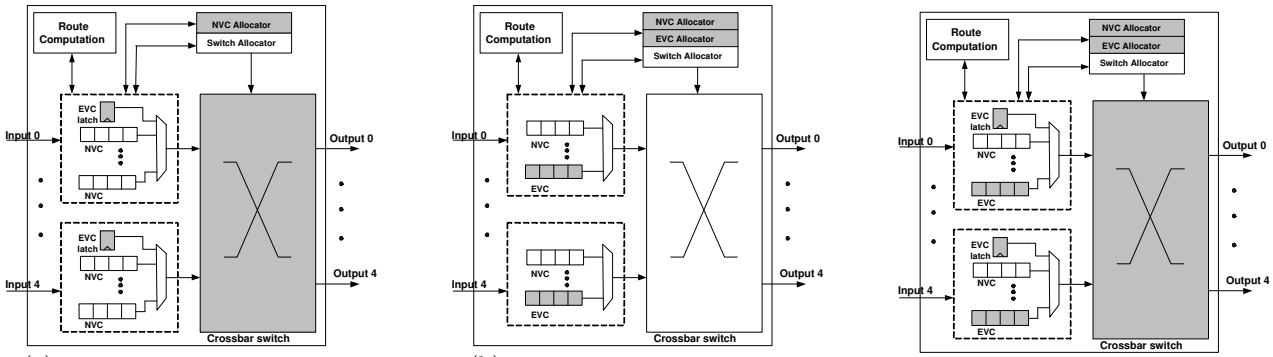
Virtual channel allocator: In order to prevent head-of-line blocking, two separate sets of VC allocators are used at every node in a dynamic EVC design and source/sink nodes in a static EVC design: one which allocates EVCs and the other NVCs. Depending on the output port and number of hops left in the packet's next dimension, the packet either places a request to the NVC allocator (if the number of hops left in the next dimension is less than the uniform EVC length for static EVCs or the smallest EVC length for dynamic EVCs) or otherwise to the EVC allocator. Bypass nodes in a static EVC design, on the other hand, allocate only NVCs and, hence, have only one VC allocator.

Switch design: For the express pipeline in Fig. 5(b), no modifications need to be made to the crossbar switch design, since the EVC latch in the input port is multiplexed with the NVC input buffers, sharing a single input port to the crossbar. However, to achieve the aggressive express pipeline in Fig. 5(c), where EVC flits bypass the switch altogether, the EVC latch needs to be physically located near the center of the router. This EVC latch acts as a staging latch, breaking the flit's path through the network and effectively bypassing the entire data-path of the router.

Buffer management circuit: Fig. 10 shows the block diagram for dynamically-managed buffers. At each input port, a router maintains a pool of free flit buffers. When a new flit arrives, it is assigned the buffer slot at the head of this pool in the BW pipeline stage along with storing the assigned buffer slot in the packet control block. The function of the packet control block is to store the buffer slot pointers for tracking the buffers assigned to all flits corresponding to different input VCs. This pointer information is used to read

(a) Static EVCs (b) Dynamic EVCs

Figure 8: Starvation scenario



(a) Bypass node in static EVC network (b) Source/sink node in static EVC network (c) Dynamic EVC network node

Figure 9: EVC router microarchitecture

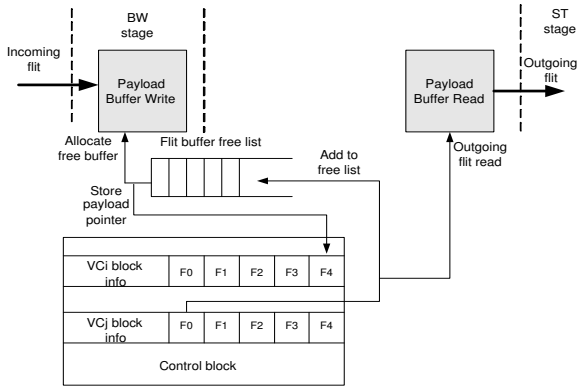


Figure 10: Dynamic buffer management circuit

out the flit from the buffer pool before it traverses the switch in the ST stage, after which the buffer slot is returned to the free pool. Since the pointer value can be prefetched before the ST stage begins, it does not add to the critical path. Apart from this, the buffer slot occupied by the header of every packet is marked as reserved for that packet and is not added to the free pool until the tail of that packet leaves the router. This is done to prevent deadlocks and ensure forward progress within each packet.

Reverse signaling overhead: As compared to the baseline network, an EVC-based design requires extra reverse wiring between nodes which are virtually connected for flow control and starvation signaling. This overhead can be broken into the following categories:

VC signaling: These wires are used to signal availability of VCs across nodes. For a baseline design with v VCs per port, the number of wires at any cross-section required for VC signaling along a particular direction, $W_{vc\_signaling}$, is given by:

$$W_{vc\_signaling} = \lceil log_2 v \rceil \quad (11)$$

On the other hand, the corresponding $W_{vc\_signaling}$ for a static EVC design with v VCs partitioned into n NVCs and e EVCs, is given by:

$$W_{vc\_signaling} = \lceil log_2 n \rceil + \lceil (\log_2 e) \rceil \quad (12)$$

For a dynamic EVC network with a maximum EVC length of l_{max}, communication of VC availability has to be done across l_{max} nodes on either side. In this case, $W_{vc\_signaling}$ is given by:

$$W_{vc\_signaling} = \lceil log_2 n \rceil + \left\lceil (\log_2 m).\frac{(l_{max}).(l_{max}-1)}{2} \right\rceil \quad (13)$$
$$m = e/(l_{max}-1)$$

where m is the number of VCs for EVCs of a particular length (assuming all e EVCs are uniformly partitioned among EVCs with a length of two through l_{max} hops).

Buffer threshold signaling: To signal buffer availability using on/off threshold tokens, the baseline network requires a single wire in each direction connecting adjacent nodes while a static EVC design requires two wires (one connecting adjacent nodes and the other connecting to the node with which a fixed-length EVC connection exists). On the other hand, in a dynamic EVC design, each node needs to communicate buffer availability to l_{max} neighbors. We use 1-hot decoded wires for this purpose, giving a wiring overhead $W_{thr\_signaling}$ in a particular direction of:

$$W_{thr\_signaling} = l_{max} \quad (14)$$

Starvation signaling: Again, while a single wire between any pair of virtually connected nodes is sufficient for starvation signaling in static EVCs and dynamic EVCs with $l_{max} = 2$, in general a dynamic EVC network requires starvation tokens to be communicated to $l_{max} - 1$ neighbors (as explained in Section 3.4) giving a starvation wiring overhead $W_{starvation}$ of:

$$W_{starvation} = \lceil log_2(l_{max} - 1) \rceil \quad (15)$$

Hence, it can be seen that while the baseline and static EVCs have comparable constant reverse signaling overhead, this overhead for dynamic EVCs is dependent on l_{max}, with $W_{vc\_signaling}$ increasing quadratically while $W_{thr\_signaling}$ and $W_{starvation}$ having a linear and logarithmic dependence on l_{max}, respectively. This wiring overhead as a percentage of the forward wiring (assuming 128-bit wide channels and eight VCs per port which are uniformly partitioned between all VC bins) is 3% (4 wires) for the baseline, 5% (7 wires) for the static EVC design, 7% (9 wires) for a dynamic EVC design with $l_{max} = 2$, 14% (18 wires) for $l_{max} = 3$ and 20% (26 wires) for $l_{max} = 4$. Hence, it can be seen that the reverse wiring for a dynamic EVC network with $l_{max} = 2$ is comparable to the baseline and static EVCs and is a small fraction of the forward flit-wide wiring. This overhead, however, increases slowly with higher values of l_{max}.

5. EVALUATION

In this section, we present a detailed evaluation of EVCs, exploring its design space while comparing it against the state-of-the-art baseline design (BASE+LA+BY+SP) and ideal interconnect yardsticks described in Section 2.

5.1 Simulation infrastructure

To model network performance, we use a cycle-accurate micro-architecture simulator. The model may be stimulated via a) a synthetic traffic generator mode or b) in trace

mode using traffic traces. The network models all major components of the router pipeline at clock granularity, viz., buffer management, routing algorithm, VC and switch allocation and flow control between routers, with each component's circuit schematics designed and pipeline stages sized via detailed critical path analysis (see Section 2). We use Orion [19], an architecture-level network energy model, to evaluate the energy consumption of routers and links. Orion includes detailed dynamic and leakage energy models [20] for all major router microarchitecture components, including input buffers, crossbar, arbitration logic as well as network links.

Table 2 lists the EVC-specific parameters used in this study, while Table 1 lists the general process parameters and that of the baseline network. The entire set of VCs in an EVC network is partitioned into NVCs and EVCs. For static EVCs, the entire EVC set acts as a contiguous pool whereas they are further partitioned into $l_{max} - 1$ bins in case of dynamic EVCs (Section 3.2). For better utilization of buffers, we use a dynamically-managed design (Section 3.3), with the entire buffer set at any router port shared among all VCs. The buffer size is kept the same as that in the baseline design. Unless otherwise mentioned, all EVC experiments use the aggressive EVC pipeline with an EVC length of two hops for static EVCs. To keep the reverse wiring overhead comparable to the baseline, an l_{max} of two hops is also used for dynamic EVCs.

Evaluation using synthetic traffic uses packets which uniformly consist of two sizes: single-flit short packets and long packets consisting of five flits. Each simulation run for a synthetic traffic is 1 million cycles long. For all runs, a warm-up period of 100k cycles is assumed. Network saturation is defined as the point at which packet latency is three times the zero-load latency.

The SPLASH-2 [14] traces were gathered by running the benchmarks on Bochs [21], a multiprocessor simulator with an embedded Linux 2.4 kernel. Each benchmark was run in Bochs with N concurrent threads, where N is the size of the chip/network, and the memory trace captured. This memory trace is then applied to a memory system simulator that models the classic MSI (Modified, Shared, Invalid) directory-based cache coherence protocol, with the home directory nodes statically assigned based on the least significant bits of the tag, distributed across all processors in the entire chip. Each processor node has a two-level cache (2MB L2 cache per node) that interfaces with a network router and 4GB off-chip main memory. Access latency to the L2 cache is derived from Cacti to be six cycles, whereas off-chip main memory access delay is assumed to be 200 cycles.

5.2 Uniform random traffic

Uniform random traffic assumes each node uniformly injects packets into the network with randomly distributed destinations. This highlights the best-case throughput for the baseline network. Here, we use uniform traffic to present a comparison of EVCs against the baseline. In all results, percentage improvements are reported with respect to the baseline.

Performance evaluation: Fig. 11(a) plots flit latencies for uniform random traffic for a 7×7 network as a function of the injected load. EVC lengths are assumed to be two hops. Clearly, the EVC-based design outperforms the baseline in terms of throughput as well as latency under all levels of network traffic. The latency reduction of static EVCs as compared to the baseline is 29.2% before the baseline saturates whereas the corresponding reduction due to dynamic EVCs is 44.7%. Moreover, dynamic EVCs also lead to a significant improvement in throughput with the network saturating at around 82% of network capacity.

It can be seen that when the network load is low, both EVCs and baseline perform well as packets are able to use pipeline bypassing to shorten the critical path. However, as the load increases, the latency gap between the baseline and EVCs widens. Although the baseline relies on aggressive speculation to reduce delay, such techniques fail more often and show a diminishing advantage with increasing traffic and contention. EVCs, on the other hand, are able to sustain low latencies with increasing traffic by creating virtual dedicated wires in the network, allowing packets to bypass intermediate nodes along their path (and hence reduce latency) in a completely *non-speculative* manner. Moreover, EVCs lower the average per-hop contention seen by a packet, thereby pushing throughput. In trying to analyze this further, we see that dynamic EVCs perform better than static EVCs by increasing the usage of the virtual lanes created by EVCs and, hence, increasing the average number of nodes a packet bypasses, the latency reduction of dynamic EVCs over the static EVC design being around 67% before the static design saturates. It can be seen that using EVCs, the no-load latency is reduced to 14.5 cycles (approaching the 11-cycle ideal interconnect delay).

Energy evaluation: Fig. 11(b) shows the normalized router energy consumption at 70% capacity (before the baseline saturates) for a 7×7 mesh network. As can be seen, EVCs are able to significantly reduce overall router energy. This reduction is 21% for static EVCs and 24.5% for dynamic EVCs as compared to the baseline. This is mainly due to a reduction in buffer energy [by 25% (30%) for static (dynamic) EVCs] and crossbar energy [by 29% (33%) for static (dynamic) EVCs] due to bypassing of intermediate nodes when using EVCs. Fig. 11(c) highlights this fact by comparing the energy of different router components. Arbitration energy (which is mostly dominated by control logic) was found to be insignificant and is omitted from the figure. When compared with a baseline without the straight-through buffers and cut-through crossbar energy optimizations discussed in Section 2.1, the energy reduction using power-optimized dynamic EVCs increases to 53.4%.

It should be noted that our default parameters assume equal network buffer capacity for both EVCs and the baseline. Although EVCs achieve a much higher throughput using this configuration, the designer can make an energy-performance trade-off by reducing the buffer size for EVCs and, hence, lowering both dynamic and leakage energy at the expense of some loss in throughput.

5.3 Impact of different express pipelines

When using the non-aggressive express pipeline, where bypassing EVC flits need to go through switch traversal, improvements in packet energy and delay diminish by a small amount. Fig. 12 plots packet latency as a function of network load for a 7×7 mesh, with EVCs using a non-aggressive pipeline and assuming uniform random traffic. In case of dynamic EVCs, the non-aggressive design leads to a 11.6% higher latency near saturation as compared to the aggressive pipeline. The router energy consumption goes up by 8% mainly due to an increase in crossbar energy.

5.4 Impact on network contention

By allowing packets to bypass nodes virtually, EVCs ease contention for resources in the network. To study this effect, Fig. 13 compares the average contention delay, or the time which packets spend waiting for network resources, near saturation for the different designs in a 7×7 network. It can be seen that EVCs significantly reduce the level of contention in the network with the contention delay reducing by 28.5% (47%) for static (dynamic) EVCs. This is because EVC flits do not need to win switch ports and VCs while bypassing a node, as they are prioritized to directly use the switch port, which is equivalent to a 100% resource allocation efficiency while bypassing. This reduction in contention for network resources translates directly to a higher throughput.

5.5 Effect of a larger network

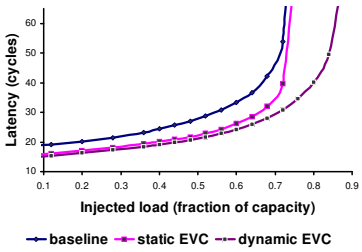
To study the scalability of EVCs, we consider a 10×10 network with longer EVC lengths of three hops for static EVCs and $l_{max} = 3$ for dynamic EVCs (assuming uniform VC partitioning between EVCs of different lengths and the aggressive express pipeline). Fig. 14(a) plots packet latency as a function of network traffic (assuming uniform random traffic) for this network. As can be seen, EVCs continue to show a considerable performance gain as compared to the baseline, with the reduction in latency being 34.4% for static EVCs and 52.8% for dynamic EVCs just before the baseline

Table 1: Baseline process and network parameters

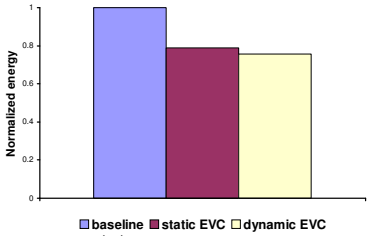
Technology	65 nm
V_{dd}	1.1 V
$V_{threshold}$	0.17 V
Frequency	3 GHz
Topology	7-ary 2-mesh
Routing	Dimension-ordered (DOR)
Traffic	Uniform random
Number of router ports	5
VCs per port	8
Buffers per port	24
Flit size/channel width (c_{width})	128 bits
Link length	1 mm
Wire pitch (W_{pitch})	0.45μm

Table 2: EVC-specific parameters

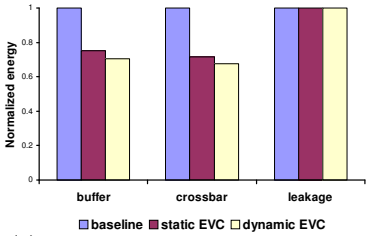
EVC pipeline	Aggressive express pipeline
Buffer management	dynamic
Buffers per port	24
Static EVC-specific parameters	
EVC length	2 hops
NVCs per port	4
EVCs per port	4
Dynamic EVC-specific parameters	
l_{max}	2
NVCs per port	2
EVCs per bin	6
Starvation-avoidance parameters	
n	20
p	3



(a) Network performance



(b) Router energy



(c) Energy saving from different router components

Figure 11: Uniform random traffic results

saturates. Moreover, the throughput for dynamic EVCs approaches 88% of network capacity (23% improvement over the baseline). This is mainly attributable to the fact that with longer and flexible EVC lengths, packets are able to bypass more hops along their path. Fig. 14(b) shows the normalized router energy at 75% capacity (before the baseline saturates). In this case, EVCs show a more pronounced reduction in energy, with the average router energy lowered by 23.5% for static EVCs and 38% for dynamic EVCs. Again, this is mainly attributable to a reduction in buffer energy [reduced by 29% (47%) using static (dynamic) EVCs] and crossbar energy [reduced by 32% (50%) for static (dynamic) EVCs]. However, as explained before, a network with l_{max} = 3 incurs a slightly higher reverse wiring overhead as compared to the baseline.

5.6 Dynamic EVC design space

In this section, we explore the design space of dynamic EVCs focusing in particular on the routing flexibility in using EVCs and trade-offs related to EVC lengths.

EVC routing flexibility: When using a dynamic EVC design with $l_{max} > 2$, performance can be further increased by allowing flexibility in choosing EVCs, as explained in Section 3.2. This is especially true for non-uniform traffic, as shown in Fig. 15, which plots latency as a function of injected load for a 7×7 network, assuming the shuffle traffic pattern. l_{max} is chosen to be four hops, thereby allowing packets to choose between EVCs of lengths two, three and four hops when route flexibility is used. The partitioning of VCs is made non-uniform with a higher number of VCs given to smaller EVC lengths. Out of the total of eight VCs per port, two are assigned to NVCs whereas the rest are divided as three VCs, two VCs and one VC for the two-, three- and four-hop EVC bins, respectively. As compared to a design without route flexibility, a latency reduction of 26% is seen near saturation along with a slight improvement in throughput. This is mainly attributable to a significant lowering in contention for VCs due to the spreading of packets among different virtual paths owing to the flexibility in choosing EVCs of different lengths.

Maximum EVC length: We next explore the trade-offs related to the maximum EVC length l_{max} in a dynamic EVC network. Assuming a 7×7 network using the aggressive express pipeline and uniform random traffic, the no-load latency was found to be 14.5 cycles for l_{max} = 2, 13.6 cycles for l_{max} = 3 and 13.2 cycles for l_{max} = 4 with the saturation throughput as a fraction of capacity being 82%, 84% and 86% for l_{max} = 2, 3 and 4, respectively. It can be seen that larger values of l_{max} lead to better performance with the no-load latency of the l_{max} = 4 network approaching the 11 cycles ideal interconnect latency. This is because longer EVCs allow packets to bypass more nodes along their paths (shown in Fig. 16(a)). However, as can be seen, the gain in performance is not proportional to the increase in the number of nodes bypassed. Moreover, increasing l_{max} comes at the cost of a higher reverse wiring overhead, as explained in Section 4. In trying to study this further, we analyze the average contention delay T_c, or the time which packets spend waiting at intermediate routers near the saturation point (shown in Fig. 16(b)). It can be seen that whereas T_c reduces with increasing l_{max}, leading to lower packet latencies, not all components of T_c decrease. Increasing l_{max} significantly lowers switch contention or the time spent waiting to win switch ports at intermediate hops due to a higher number of nodes bypassed and, hence, fewer switch allocations which a packet has to go through. However, as dynamic EVCs lead to a partitioning of the entire set of VCs at a port into l_{max} bins, a larger l_{max} implies more bins with fewer VCs per bin (as the total number of VCs is kept constant). Hence, the contention for VCs within a bin increases, leading to an increase in the delay due to VC contention, assuming routing flexibility among EVCs is not used. Moreover, the number of EVC paths bypassing a node increases in proportion to l_{max}, leading to a higher probability of a flit waiting at a router because of a bypassing EVC flit using the output link (shown as wait delay). As EVCs do not add extra physical channels, this wait delay is unavoidable because the same physical channel is shared between waiting and bypassing flits. It should be noted, however, that we found T_c to be mainly dominated by switch and VC contention delays, with wait delay contributing only a small fraction.

Fig. 16 also highlights the effect of routing flexibility (in a network with l_{max} = 4) on T_c. Packets use routing flexibility to switch to smaller-length EVCs if longer ones are busy, which causes the number of bypassed nodes to decrease slightly with a corresponding increase in switch contention. However, the contention for VCs is reduced significantly due to the distribution of packets among paths of different EVC lengths based on the relative contention within each EVC bin, leading to a reduction in the overall contention delay.

5.7 SPLASH results

In this section, we present evaluation results using benchmarks from the SPLASH-2 [14] suite for a 7×7 network, assuming an EVC length of two hops for static EVCs and l_{max} = 2 for dynamic EVCs (with the aggressive express pipeline). Fig. 17(a) shows the comparison of normalized delay for each benchmark against the baseline. It can be seen that EVCs show a latency improvement over the base-

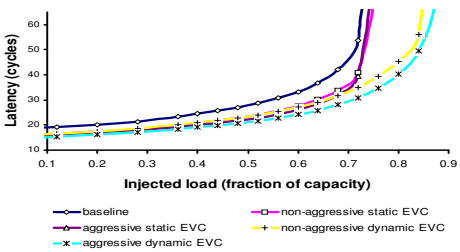


Figure 12: Impact of different express pipelines on latency

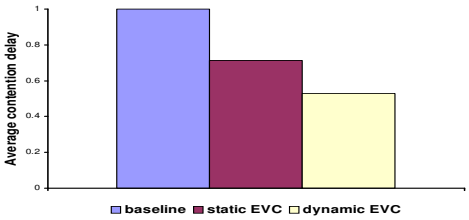
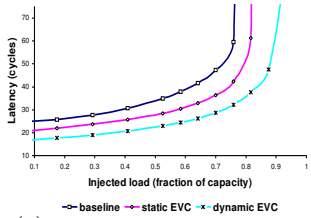
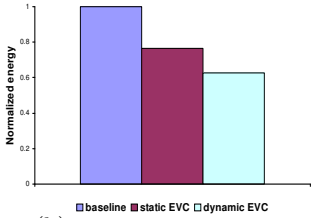


Figure 13: Relative contention for network resources



(a) 10x10 network performance



(b) 10x10 network energy

Figure 14: EVCs in a larger network

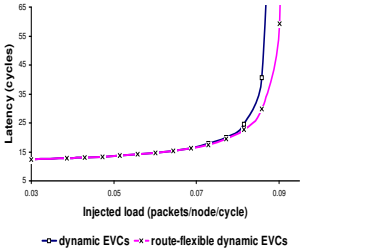
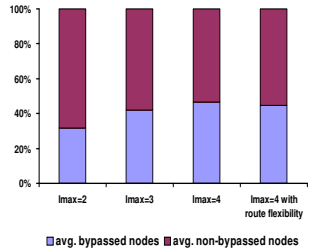
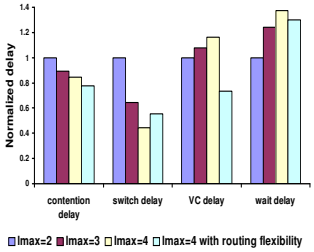


Figure 15: Impact of routing flexibility when using EVCs

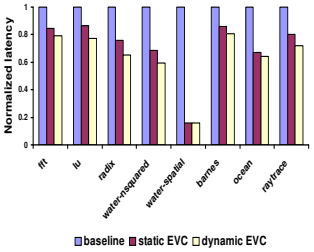


(a) Comparison of bypassed nodes

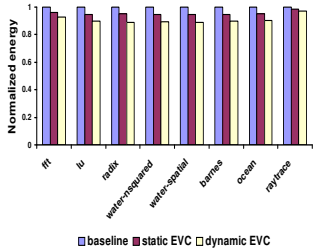


(b) Packet contention delay analysis

Figure 16: Analysis with varying l_{max}



(a) Normalized latency



(b) Normalized energy

Figure 17: SPLASH results

line for all benchmarks. The largest reductions were seen for *water-spatial* and *water-nsquared*, with static EVCs reducing latency by 84% (31.3%) for *water-spatial* (*water-nsquared*), the corresponding reductions for dynamic EVCs being 84% (40.3%). This is mainly attributable to high traffic levels in these benchmarks where the speculative techniques used by the baseline fail whereas EVCs, being non-speculative, show large gains. On the other hand, *fft, lu, radix, barnes* and *ocean* represent low to medium traffic with moderate hop-counts, leading to a latency reduction of 20.8%, 22.9%, 34.6%, 19.4% and 35.8%, respectively, using the dynamic EVC design. *Raytrace*, on the other hand, represents very low traffic but with a high hop-count. Hence, whereas the baseline performs well due to pipeline bypassing and speculation, packet latencies are reduced further using EVCs (by 28.2% using dynamic EVCs) due to their higher utilization and the use of aggressive switch bypassing.

Fig. 17(b) shows the normalized router energy for the different benchmarks. Again, EVCs show energy gains, the reduction using dynamic EVCs being around 9% on average going upto 11% for *water-spatial*.

6. RELATED WORK

Topological and routing techniques. Substantial prior work has explored alternative topologies for reducing network latency through lowering of hop-count [10]. Higher-radix topologies lead to a lower hop-count, but present challenges in router design [22]. Numerous routing algorithms have also been proposed [23,24]. These are all orthogonal to EVCs, which target per-hop latency and energy rather than the hop-count. EVCs can complement any topology and routing algorithm to further drive network latency, throughput and energy towards the ideal. However, EVCs are inspired by the express cubes topology [11]. In express cubes, extra physical links that span multiple hops are proposed to allow packets to skip intermediate router pipelines, and shown to lead to significant energy savings when the upper metal interconnect is used for such metal links in on-chip networks [17]. EVCs essentially *virtualize* the physical express links of express cubes, thereby obviating the need for the extra metal interconnects and highly-ported routers at the sources and sinks of express links. Smaller number of ports directly translates to smaller router area footprint and lower energy consumption. Virtualizing these physical express links also means that there will not be wastage of physical link bandwidth when the traffic pattern is adversarial. In addition, EVCs lead to a higher reduction in packet latencies as compared to express cubes under low loads when network contention is low. If the total amount of wiring is assumed to be constant, an express cube design would lead to longer packet sizes due to reduced wiring per channel. More specifically, a flat express cube, with an interchange every k nodes, would lead to doubling of packet sizes, giving a latency for a packet traveling H hops (assuming no contention and equal router and interchange delays which favors express cubes, and that $k \mid H$) to be $T = D/v + 2 \cdot L/b + (H/k + k) \cdot T_{router}$, whereas the corresponding latency in a k-hop static EVC design (assuming the aggressive express pipeline) is given by $T = D/v + L/b + (H/k + k - 1) \cdot T_{router}$. It can be seen that express cubes increase the serialization delay due to longer packet sizes. For hierarchical express cubes, this overhead grows directly in proportion to the number of interchange levels. Moreover, using dynamic EVCs, virtual express paths of a range of lengths can be created from every network node with a low overhead in reverse wiring as compared to the wiring overhead of a similar design with multiple physical express channels originating from every node.

Ogras et al. [25] proposed long-range link insertion to connect non-adjacent nodes in an application-aware fashion. This again requires additional flit-wide physical wiring and fatter routers with larger number of ports and hence larger

crossbar switches. In contrast, EVCs avoid these overheads by using virtual connections between distant nodes.

Microarchitectural techniques targeting T_{router} and throughput. Circuit switching first allocates channels to form a pre-reserved circuit prior to communication, so that actual communication can occur with close-to-ideal network latency, incurring just switch and link traversal latencies within the router pipeline. Researchers have also looked at hybrid flow control mechanisms, such as wave switching [26] and pipeline circuit switching [27], which combine the advantages of circuit and packet switching. However, circuit switching and its variants incur a large circuit setup time which increases latency for short transfers. Circuit switching also suffers in throughput, as links are not shared efficiently amongst different message flows. Flit-reservation flow control [28] sends out control flits in advance to schedule buffers and channels along the way for subsequent data flits. This can lower the router pipeline delay to close to the ideal while improving throughput, but requires control flits to be sent along fast upper metal wires, contending for metal layers with other global wiring. Besides, it requires a complex router microarchitecture that leads to significant hardware overhead.

Our baseline router already incorporates several speculation mechanisms that target T_{router}, such as bypassing and speculation. Mullins et al. [12] propose a doubly-speculative single-stage design in which arbitration decisions are precomputed to further reduce pipeline dependencies. While speculative designs work well under low loads, they perform poorly under heavy loads when network contention is high. Moreover, speculation often employs redundant logic which increases energy consumption. Our proposed EVCs are completely non-speculative and, hence, can improve performance at all traffic conditions while simultaneously lowering the energy consumption. Kim et al. [29] target throughput by proposing path-sensitive packet buffering and partitioning the crossbar into separate row and column modules to reduce contention. However, this scheme relies on buffering of flits based on their output path and, hence, assumes multi-ported buffers which incur significant overhead in resource-constrained on-chip designs. EVCs, on the other hand, push *both* latency and throughput towards the ideal using single-ported buffers and crossbars that are area- and energy-efficient.

7. CONCLUSION

Although the multiplexing of network channels over multiple packet flows in packet-switched on-chip network designs leads to throughput gains, this comes with a significant latency, energy and area overhead in the form of complex routers. In this paper, we target this overhead in an attempt to approach the ideal interconnection fabric of dedicated wires between all nodes. We propose EVCs, a novel flow control and router microarchitecture design, which use virtual lanes in the network to allow packets to bypass nodes along their path in a non-speculative fashion and, hence, significantly reduce delay and energy consumption. The virtual paths created by EVCs also help lower the level of contention in the network, thereby allowing the network to push through more packets before saturation and, hence, improve throughput. Since network nodes are virtually connected using existing physical channels, EVCs improve performance with a negligible wiring area overhead and minimal hardware complexity. We presented a detailed microarchitecture for EVCs, analyzing the hardware and pipeline complexity of each of its components. A detailed evaluation, using both synthetic and actual workloads, showed EVCs significantly improving network energy/delay and throughput as compared to a state-of-the-art packet-switched network, and closely approaching the ideal interconnect.

Acknowledgments

The authors would like to thank William J. Dally of Stanford University for useful feedback on this work. We would also like to thank Ted Tabe and David V. James of Intel Corp. for their help with the modeling infrastructure and comments on the microarchitecture. This work was supported in part by the MARCO Gigascale Systems Research Center, Alfred P. Sloan Research Foundation, a grant from Intel Corporation, an Intel PhD Fellowship and NSF under grant no. CNS-0613074.

References

[1] "International Technology Roadmap for Semiconductors," http://public.itrs.net.

[2] R. Ho, K. Mai, and M. Horowitz, "The future of wires," *Proc. IEEE*, vol. 89, no. 4, Apr. 2001.

[3] K. Sankaralingam *et al.*, "Exploiting ILP, TLP, and DLP with the polymorphous TRIPS architecture," in *Proc. Int. Symp. Computer Architecture*, June 2003, pp. 422–433.

[4] M. B. Taylor *et al.*, "Evaluation of the Raw microprocessor: An exposed-wire-delay architecture for ILP and streams," in *Proc. Int. Symp. Computer Architecture*, June 2004.

[5] L. Benini and G. De Micheli, "Networks on chips: A new SoC paradigm," *IEEE Computer*, vol. 35, no. 1, pp. 70–78, Jan. 2002.

[6] W. J. Dally and B. Towles, "Route packets not wires: On-chip interconnection networks," in *Proc. Design Automation Conf.*, June 2001.

[7] J. A. Kahle *et al.*, "Introduction to the Cell multiprocessor," *IBM Journal of Research and Development*, vol. 49, no. 4/5, 2005.

[8] M. Sgroi *et al.*, "Addressing the system-on-a-chip interconnection woes through communication-based design," in *Proc. Design Automation Conf.*, June 2001.

[9] W. J. Dally, "Virtual-channel flow control," in *Proc. Int. Symp. Computer Architecture*, May 1990, pp. 60–68.

[10] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers, 2004.

[11] W. J. Dally, "Express cubes: Improving the performance of k-ary n-cube interconnection networks," *IEEE Trans. on Computers*, vol. 40, no. 9, Sept. 1991.

[12] R. Mullins, A. West, and S. Moore, "Low-latency virtual-channel routers for on-chip networks," in *Proc. Int. Symp. Computer Architecture*, June 2004, pp. 188–197.

[13] L.-S. Peh and W. J. Dally, "A delay model and speculative architecture for pipelined routers," in *Proc. Int. Symp. High Performance Computer Architecture*, Jan. 2001, pp. 255–266.

[14] "SPLASH-2," http://www-flash.stanford.edu/apps/SPLASH/.

[15] M. Galles, "Scalable pipelined interconnect for distributed endpoint routing: The SGI SPIDER chip." in *Proc. Hot Interconnects 4*, Aug. 1996, pp. 141–146.

[16] S. S. Mukherjee, *et al.*, "The Alpha 21364 network architecture," *IEEE Micro*, vol. 22, no. 1, pp. 26–35, Jan./Feb. 2002.

[17] H.-S. Wang, L.-S. Peh, and S. Malik, "Power-driven design of router microarchitectures in on-chip networks," in *Proc. Int. Symp. Microarchitecture*, Nov. 2003, pp. 105–116.

[18] W. Liao and L. He, "Full-chip interconnect power estimation and simulation considering repeater insertion and flip-flop insertion," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 2003, pp. 574–580.

[19] H.-S. Wang, *et al.*, "Orion: A power-performance simulator for interconnection networks," in *Proc. Int. Symp. Microarchitecture*, Nov. 2002, pp. 294–305.

[20] X.-N. Chen and L.-S. Peh, "Leakage power modeling and optimization of interconnection networks," in *Proc. Int. Symp. Low Power Electronics and Design*, Aug. 2003, pp. 90–95.

[21] K. P. Lawton, "Bochs: A portable PC emulator for Unix/X," *Linux J.*, vol. 1996, no. 29, p. 7, 1996.

[22] J. Kim, *et al.*, "Microarchitecture of a high-radix router," in *Proc. Int. Symp. Computer Architecture*, June 2006, pp. 420–431.

[23] J. Hu and R. Marculescu, "DyAD - Smart routing for networks-on-chip," in *Proc. Design Automation Conf.*, June 2004.

[24] D. Seo, *et al.*, "Near-optimal worst-case throughput routing for two-dimensional mesh networks," in *Proc. Int. Symp. Computer Architecture*, June 2005.

[25] U. Y. Ogras and R. Marculescu, "It's a small world after all: NoC performance optimization via long-range link insertion," *IEEE Trans. Very Large Scale Integration Systems*, vol. 14, no. 7, pp. 693–706, July 2006.

[26] J. Duato, *et al.*, "A high performance router architecture for interconnection networks," in *Proc. Int. Conf. Parallel Processing*, Aug. 1996, pp. 61–68.

[27] P. T. Gaughan and S. Yalamanchili, "A family of fault-tolerant routing protocols for direct multiprocessor networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 6, no. 5, May 1995.

[28] L.-S. Peh and W. J. Dally, "Flit-reservation flow control," in *Proc. Int. Symp. High Performance Computer Architecture*, Jan. 2000, pp. 73–84.

[29] J. Kim, *et al.*, "A gracefully degrading and energy-efficient modular router architecture for on-chip networks," in *Proc. Int. Symp. Computer Architecture*, June 2006, pp. 4–15.